23CSE111

OBJECT ORIENTED PROGRAMMING

LAB MANUAL



Department of Computer and Science Engineering

Amrita School of Engineering

Amrita Vishwa Vidyapeetham, Amaravati Campus

Name:-P LIKHITH SAI

Sec:-CSE-A

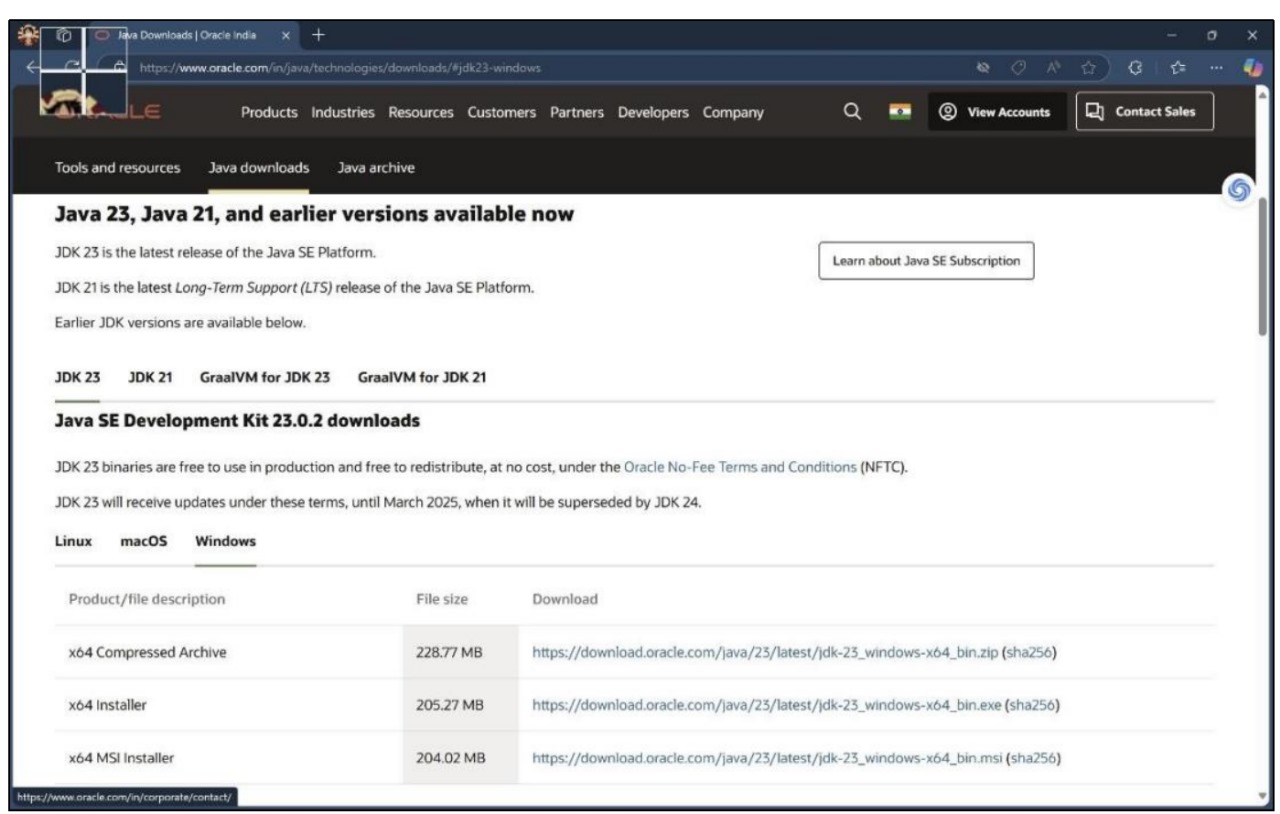
Roll.No:-AV.SC.U4CSE24306

Verified By:

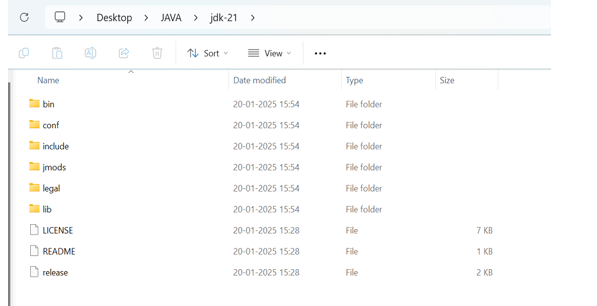
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **ACTIVITY** | **PAGE** | **DATE** | **SIGN** |
|  | **WEEK-1** |  | 27-01-2025 |  |
| 1 | Download and install JDK |  |  |  |
| 2 | Write a Java program to print the message Welcome to Java programming |  |  |  |
| 3 | Write a Java program to print the name, roll number, and section of the student |  |  |  |
|  | **WEEK-2** |  | 03-02-2025 |  |
| 1 | Java program to calculate the area of a rectangle |  |  |  |
| 2 | Java program to convert temperature from Celsius to farenheit |  |  |  |
| 3 | Java program to calculate simple intrest |  |  |  |
| 4 | Java program to find largest of three numbers using ternary operator |  |  |  |
| 5 | Java program to find the factorial of an number |  |  |  |
|  | **WEEK-3** |  | 11-02-2025 |  |
| 1 | Write a Java program with the following instructions.   1. Create a class with name car. 2. Create four attributes named car\_colour,car\_brand,fuel\_type,top speed. 3. Create three method named “Start\_Racing”,”End\_Race”.{ } 4. Create three objects named Car1,Car2,Car3. |  |  |  |
| 2 | Write a class by writing a Java program named Bank Account with two methods: “deposits and withdraw”. |  |  |  |
|  | **WEEK-4** |  | 02-03-2025 |  |
| 1 | Write a java program with class named “Book”. The class should contain various attributes such as “Title of the book , author , year of publication “. |  |  |  |
| 2 | To create a java program with class named Myclass with a static variable “Count” of “int type”, Initialized to 0 and a constant variable “pi” of type double, initialized to 3.1415 as attributes of that class Now, define a constructor for “Myclass” that increments the “Count” variable each that an object of Myclass is created. Finally, print the final values of the “Count” and “pi” variables |  |  |  |
|  | **WEEK-5** |  | 09-03-25 |  |
| 1 | Create a calculator using the operations including addition using subtraction, multiplication and division using multilateral inheritance and display the desired output. |  |  |  |
| 2 | Create a calculator using the operations including addition using subtraction, multiplication and division using multilateral inheritance and display the desired output. |  |  |  |
|  | **WEEK-6** |  | 16-03-2025 |  |
| 1 | Write a java program to create a Vehicle class with displayInfo() method , overridden in Car subclass toprovide info about carcompany , model , price ,seating and petrol. |  |  |  |
| 2 | An automated admission system that verifies student eligibility for UG and PG with different criteria. 1.UG requires minimum of 60% 2.PG requires minimum of 70% |  |  |  |
| 3 | Create a calculator class with  overloaded methods to perform  additions  1.add two integers4  2.add two double values  3.add three integers |  |  |  |
| 4 | Create a shape class with method calculateArea() that is overloaded for different shapes (eg: square, rectangle).Then create a subclass Circle that overrides calculateArea() method for Circle. |  |  |  |

***WEEK –1***

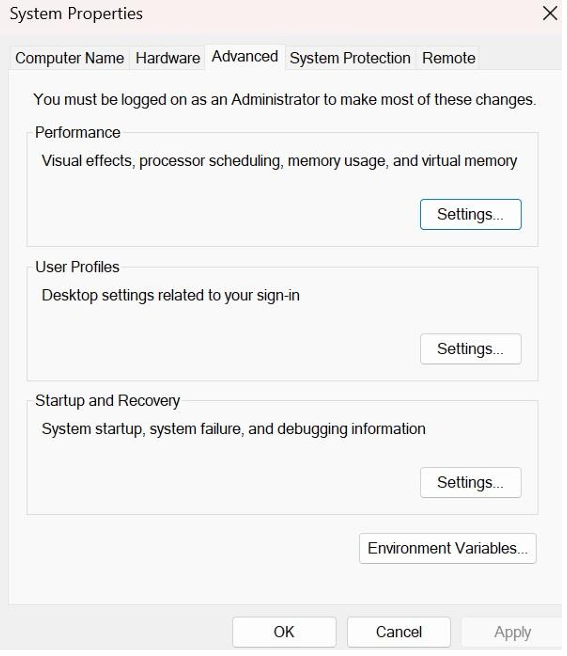
Task:-1) Write the steps to download and install Java. Aim: To download and install java. Procedure: i. Visit oracle.com website to download Java.

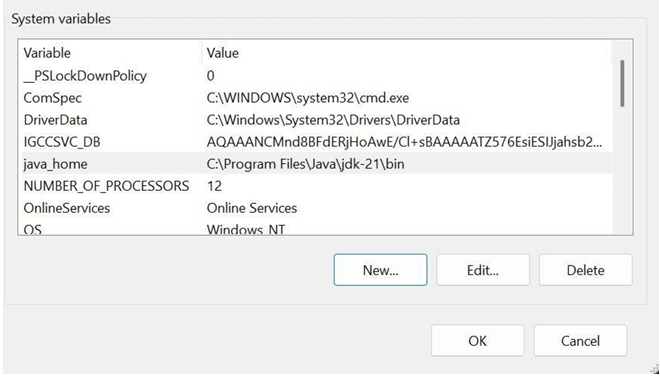


ii. Download the version which supports LTS (JDK 21) x64 installer for windows. iii. Install and copy the path.

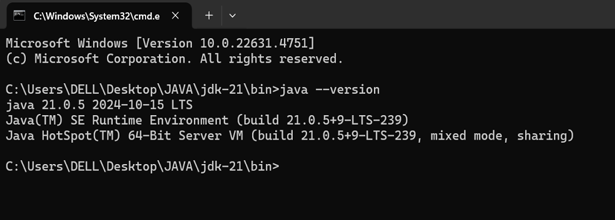


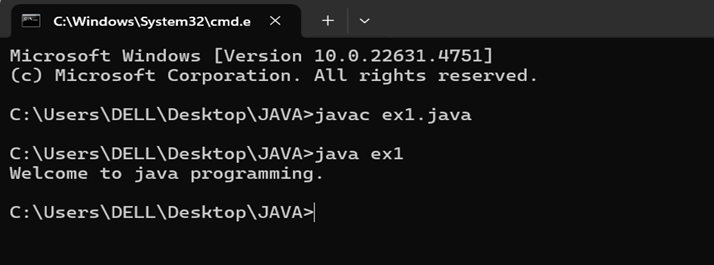
iv. Open environmental variables and add a new file with path.





v. Verify java version in command window.



Task:-2) Write a java program to print the message “Welcome to java programming”. *Code:-* public class Main { public static void main(String[] args){ System.out.println("Welcome to java programming."); } } Output:- 

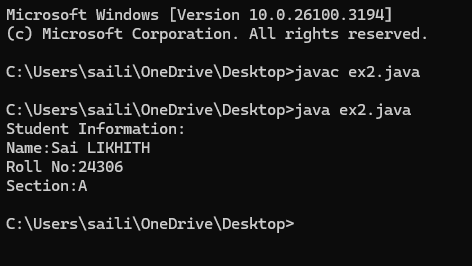
*Error:-*

|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | ; | ; is expected at end |
| 2 | S | Capital S is expected for String and System. |

Task:-3)Write a java program that prints student details(name, roll number and section of a student).

Code: class ex2{ public static void main(String[] args){ String name = "Sai LIKHITH"; int rollNo = 24306; String section = "A"; System.out.println("Student Information:"); System.out.println("Name:" + name); System.out.println("Roll No:" + rollNo); System.out.println("Section:" + section);

} } **Output:-**



|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | S | Capital S is expected for String and System. |

**WEEK-2**

**1)JAVA PROGRAM TO CALCULATE THE AREA OF THE RECTANGLE:**

**CODE:**

import java.util.Scanner;

public class area {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the rectangle: ");

double length = scanner.nextDouble();

System.out.print("Enter the width of the rectangle: ");

double width = scanner.nextDouble();

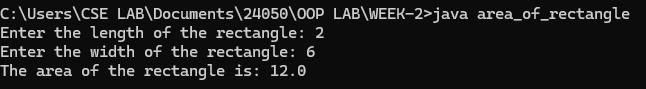
double area = length \* width;

System.out.println("The area of the rectangle is: " + area);

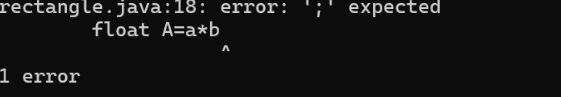
scanner.close();

}

}

**OUTPUT:**

zError:-



|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | ; | That is  semicolon( ; )  is not present at the  end of the  expression. |

**2)**

**A)JAVA PROGRAM TO CONVERT TEMPERATURE FROM CELCIUS TO FAHRENHEIT:**

**CODE:**

import java.util.Scanner;

public class celcius\_to\_fahrenheit {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter temperature in Celsius: ");

double celsius = scanner.nextDouble();

double fahrenheit = (celsius \* 9/5) + 32;

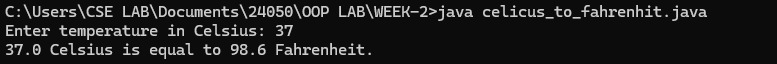
System.out.println(celsius + " Celsius is equal to " + fahrenheit + " Fahrenheit.");

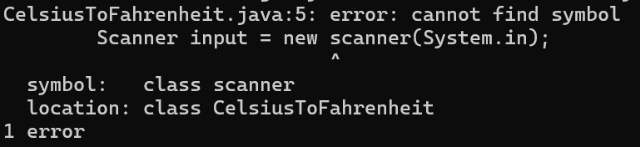
scanner.close();

}

}

**OUTPUT:**



Error:-

|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | S | Capital S is expected for String and System. |

**B)JAVA PROGRAM TO CONVERT FAHRENHEIT TO CELCIUS :**

**CODE:**

import java.util.Scanner;

public class ftoc {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter temperature in Fahrenheit: ");

double fahrenheit = scanner.nextDouble();

double celsius = (fahrenheit - 32) \* 5 / 9;

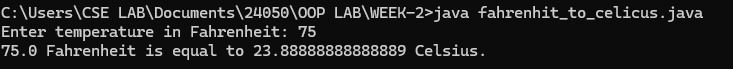
System.out.println(fahrenheit + " Fahrenheit is equal to " + celsius + " Celsius.");

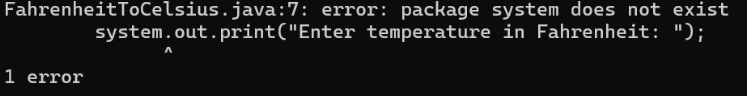
scanner.close();

}

}

**OUTPUT:**



Error:-

|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | S | Capital S is expected for String and System. |

**3)JAVA PROGRAM TO CALCULATE THE SIMPLE INTEREST:**

**CODE:**

import java.util.Scanner;

public class SimpleInterest {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the principal amount: ");

double principal = scanner.nextDouble();

System.out.print("Enter the rate of interest (in %): ");

double rate = scanner.nextDouble();

System.out.print("Enter the time period (in years): ");

double time = scanner.nextDouble();

double simpleInterest = (principal \* rate \* time) / 100;

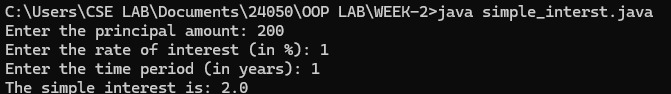
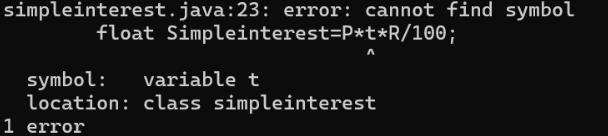
System.out.println("The simple interest is: " + simpleInterest);

scanner.close();

}

}

**OUTPUT:**

Error:-

|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | T | Capital T is expected for String and System. |

**4)JAVA PROGRAM TO FIND THE LARGEST AMONG THREE NUMBER USING TERNARY OPERATOR:**

**CODE:**

import java.util.Scanner;

public class largest3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first number: ");

int num1 = scanner.nextInt();

System.out.print("Enter the second number: ");

int num2 = scanner.nextInt();

System.out.print("Enter the third number: ");

int num3 = scanner.nextInt();

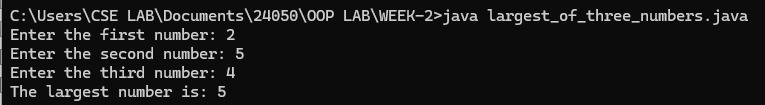
int largest = (num1 >= num2 && num1 >= num3) ? num1 : (num2 >= num1 && num2 >= num3) ? num2 : num3;

System.out.println("The largest number is: " + largest);

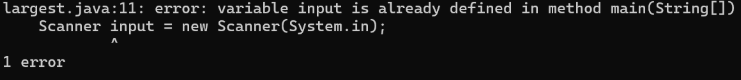
scanner.close();

}

}

**OUTPUT:**

**Error:-**



|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | main(String []). | Variable input is already defines in method main(String []). |

**5)JAVA PROGRAM TO FIND FACTORIAL OF A NUMBER:**

**CODE:**

import java.util.Scanner;

public class factorial {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number to find its factorial: ");

int number = scanner.nextInt();

long factorial = 1;

if (number < 0) {

System.out.println("Factorial is not defined for negative numbers.");

} else {

for (int i = 1; i <= number; i++) {

factorial \*= i;

}

System.out.println("The factorial of " + number + " is: " + factorial);

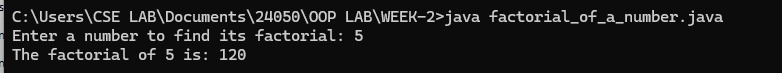
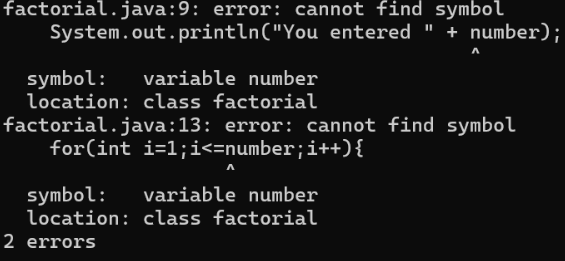
}

scanner.close();

}

}

**OUTPUT:**

Error:- 

WEEK – 3

1) Create the java program with the following instructions i) Create a class with name Car ii) Create 4 attributes named Car\_Color , Car\_brand, fuel\_type, mileage iii) Create 3 method named Start( ) , Stop( ), Service( ) iv) Create 3 objects Car1 , Car2 , Car3 v) Create a constructor which should print “Welcome to Car Garage” public class Car{

    public String carColor;

    private String carBrand;

    private String fuelType;

    public int mileage;

    Car(String carColor , String carBrand , String fuelType , int mileage){

    this.carColor =  carColor;

    this.carBrand = carBrand;

    this.fuelType = fuelType;

    this.mileage = mileage;

    System.out.println(carColor + " " + carBrand + " " + fuelType + " " +

mileage);

    }

    public void Start(){

    System.out.println("The car has just started");

    }

    public void Stop(){

    System.out.println("The car has just stopped");

    }

    public void Service(){

    System.out.println("The car is in good condition");

     }

    public static void main(String[] args){

   Car Car1 = new Car("Black","BMW","Petrol",20);

   Car Car2 = new Car("White","AUDI","Diesel",17);

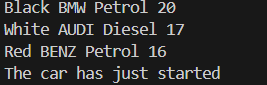
   Car Car3 = new Car("Red","BENZ","Petrol",16);

   Car1.Start();

   }

   }

OUTPUT:-



*Error:-*

|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | } | } is expected at end of the class |
| 2 | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |

Class Diagram:

|  |
| --- |
| Car |
| + carColor : String - carBrand : String - fuelType : String + mileage : int |
| + Car() : void + Start( ) : void + Stop( ) : void + Service( ) : void |

Important points:

* Private is an access specifier , It means attributes can be used in particular class only.

Method is a part of code which only runs when it is called

2)Write a java program to create a class BackAccount with two methods deposit( ) and withdraw( ) i) In deposit( ) whenever an amount is deposited it has to be updated with current amount ii) In withdraw( ) whenever an amount is withdrawn it has to be less than current amount else print “Insufficient funds”. Code: - class BankAccount{

    private String Name;

    private int AccNo, CurrBal ;

    BankAccount(String Name, int AccNo, int CurrBal){

this.Name = Name;

this.AccNo = AccNo;

this.CurrBal = CurrBal;

System.out.println("The customers are : " + this.Name + " ");

}

public int deposit(int dAmt){

CurrBal = CurrBal + dAmt ;

return CurrBal;

}

public void withdraw(int wAmount){

if(wAmount < CurrBal){

CurrBal = CurrBal - wAmount ;

System.out.println(CurrBal);

}

else{

System.out.println("Insufficient funds");

} }

public static void main(String[] args){

BankAccount likhith = new BankAccount("likhith",1500,10000);

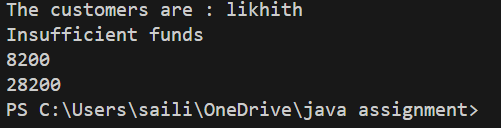
likhith.withdraw(25000);

likhith.withdraw(1800);

int FinalAmount =likhith.deposit(20000);

System.out.println(FinalAmount);

} }

OUTPUT:- 

*Error:-*

|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | } | } is expected at end of the class |
| 2 | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |

Class Diagram:-

|  |
| --- |
| BankAccount |
| - Name : String - AccNo : String - CurrBal: String |
| + BankAccount( ) : void + deposit( ) : int + withdraw( ) : void |

**Important points:**

* The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.
* The name of the constructor must match with the name of the class.
* The constructor doesn’t have a return type.

WEEK4

**1)Aim:** Write a java program with class named “Book”. The class should contain various attributes such as “Title of the book , author , year of publication “.

**CODE:**

public class Book{

// atributes

private String title;

private String author;

private int yearofpublication;

// constructor

public Book( String title, String author, int yearofpublication ){

this.title= title;

this.author= author;

this. yearofpublication= yearofpublication;

}

// method to display the details of the book

public void displaydetails(){

System.out.println("the title of the book is:"+ title);

System.out.println("the autor of book is:"+ author);

System.out.println("year of publication:"+ yearofpublication );

}

// Main method to create and display details of two books

public static void main(String[] args){

// Creating two Book objects

Book book1= new Book("Solo levelling", "p.likhith", 2025);

Book book2= new Book("attack on titan", "p.likhith", 2026);

System.out.println("book 1 Details:");

book1.displaydetails();

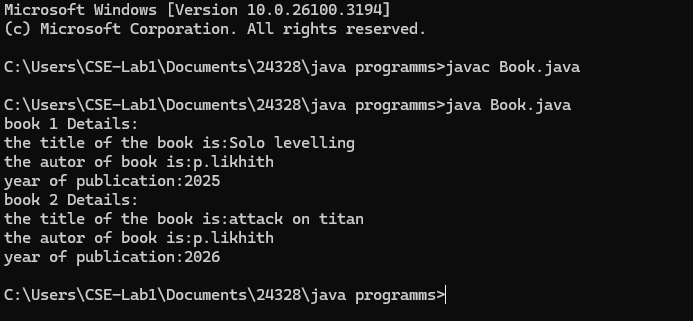
System.out.println("book 2 Details:");

book2.displaydetails();

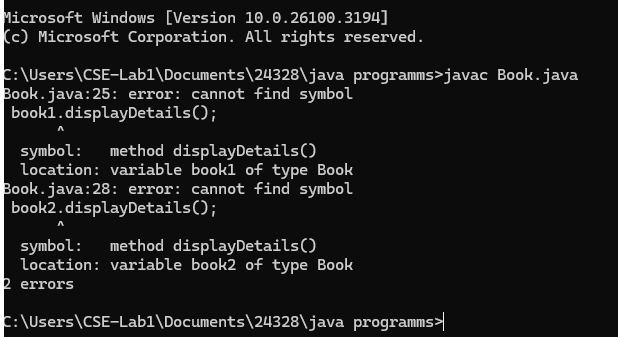
}

}

OUTPUT:



ERRORS:



**CLASS DIAGRAM**:

|  |
| --- |
| **Book** |
| - title: String  - author: String  - yearOfPublication: int |
| + Book(title: String, author: String,  yearOfPublication: int)  + displayDetails(): void |

**Important points:**

1.Declared a ‘static’ variable ‘count’ to keep the track of the number of objects are created static modifier indicates that the variable is a class level variable.

2.Declared a ‘final’ variable to ‘PI’ to represent a constant value ‘final’ modifier indicates that the variables value cannot be changed after it is initialized.

2) **AIM:** To create a java program with class named Myclass with a static variable “Count” of “int type”, Initialized to 0 and a constant variable “pi” of type double, initialized to 3.1415 as attributes of that class Now, define a constructor for “Myclass” that increments the “Count” variable each that an object of Myclass is created. Finally, print the final values of the “Count” and “pi” variables.

**CODE:**

public class MyClass {

// Static variable count of type int, initialized to zero

static int count = 0;

// Constant variable pi of type double, initialized to 314.15

final double pi = 314.15;

// Constructor for MyClass

public MyClass() {

// Increment the count variable each time an object is created

count++;

}

public static void main(String[] args) {

// Creating objects of MyClass

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

MyClass obj3 = new MyClass();

// Print the final value of count and pi

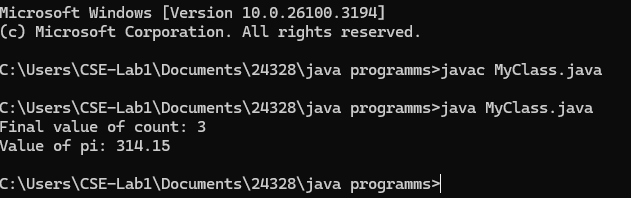
System.out.println("Final value of count: " + count);

System.out.println("Value of pi: " + obj1.pi); // Pi is the same for all objects

}

}

OUTPUT:



**CLASS DIAGRAM:**

|  |
| --- |
| **MyClass** |
| - Count: int  + pi: double |
| + MyClass()  + getCount(): int |

Important points:

1.Constructors are special methods used to initialize objects.They have the same name as the class and no return type.It can have parameters to initialize the object with specific values.Here book is a constructor.

2.Methods are used to encapsulate functionality within a class.

**WEEK 5**

**1)Aim:** Create a calculator using the operations including addition using subtraction, multiplication and division using multilateral inheritance and display the desired output.

**CODE:**

class addition{

    int x,y;

    int sum;

    public void sum(){

        sum = x + y;

        System.out.println("The sum of the numbers is: "+sum);

    }}

class subtraction extends addition{

    int diff;

    public void difference(){

        diff = x - y;

        System.out.println("The sum of the numbers is: "+diff);

    }}

class multiplication extends subtraction{

    int multi;

    public void multi(){

        multi = x\*y;

        System.out.println("The product of the numbers is: "+multi);

    }}

class division extends multiplication{

double div;

    public void div(){

        div = x/y;

        System.out.println("The division of the numbers is: "+div);

    }}

class calculator extends division{

    public calculator(int x,int y){

        this.x=x;this.y=y;

    }}

class week51{

    public static void main(String args[]){

        System.out.println("P.SAI LIKITH,24306, CSE-A");

        calculator c = new calculator(10,5);

        c.sum();

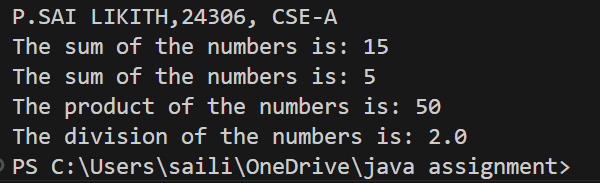
        c.difference();

        c.multi();

        c.div();

    }}

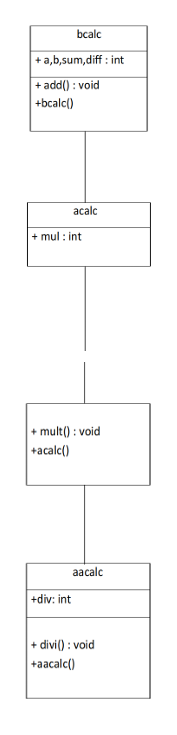
**OUTPUT:**

****

ERRORS:-

|  |  |  |
| --- | --- | --- |
| S.No. | Expected Error | Reason |
| 1 | .variable | We must mention variable name to call the variable |
| 2 | static | Static variables contain only one value |

CLASS DIAGRAMS:-



**2)AIM:**.

**CODE:**  class Vehicle {

   String brand;

   int speed;

   Vehicle(String brand, int speed) {

   this.brand = brand;

   this.speed = speed;

   }

   void displayDetails() {

   System.out.println("Brand: " + brand);

   System.out.println("Speed: " + speed + " km/h");

   }

   void startVehicle() {

   System.out.println(brand + " is starting...");

   }

   }

   // Car subclass

   class Car extends Vehicle {

   int noOfDoors;

   int seatingCapacity;

   Car(String brand, int speed, int noOfDoors, int seatingCapacity) {

   super(brand, speed);

   this.noOfDoors = noOfDoors;

   this.seatingCapacity = seatingCapacity;

   }

   @Override

   void displayDetails() {

   super.displayDetails();

   System.out.println("Number of Doors: " + noOfDoors);

   System.out.println("Seating Capacity: " + seatingCapacity);

   }

   }

   // Bike subclass

   class Bike extends Vehicle {

   boolean hasGears;

   Bike(String brand, int speed, boolean hasGears) {

   super(brand, speed);

   this.hasGears = hasGears;

   }

   @Override

   void displayDetails() {

   super.displayDetails();

   System.out.println("Has Gears: " + (hasGears ? "Yes" : "No"));

   }

   }

   // Truck subclass

   class Truck extends Vehicle {

   double capacity;

   Truck(String brand, int speed, double capacity) {

   super(brand, speed);

   this.capacity = capacity;

   }

   void showTruckDetails() {

   System.out.println("Truck Capacity: " + capacity + " tons");

   }

   @Override

   void displayDetails() {

   super.displayDetails();

   showTruckDetails();

   }

   }

   // Main class

class VehicleRentalSystem {

   public static void main(String[] args) {

   Car car = new Car("Toyota", 150, 4, 5);

   Bike bike = new Bike("Yamaha", 120, true);

   Truck truck = new Truck("Volvo", 100, 15.5);

   System.out.println("Car Details:");

   car.displayDetails();

   car.startVehicle();

   System.out.println();

   System.out.println("Bike Details:");

   bike.displayDetails();

   bike.startVehicle();

   System.out.println();

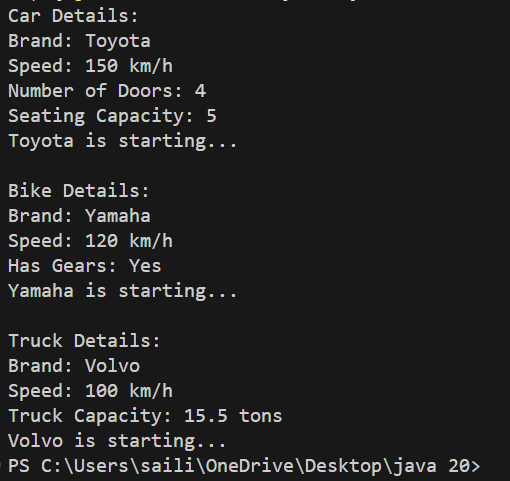
   System.out.println("Truck Details:");

   truck.displayDetails();

   truck.startVehicle();

   }}

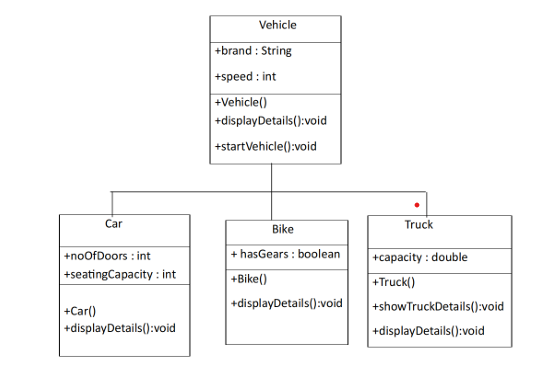
**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| S.No. | Expected Error | Reason |
| 1 | .variable | We must mention variable name to call the variable |
| 2 | static | Static variables contain only one value |

**CLASS DIAGRAMS:**



**WEEK6**

1. Write a java program to create a Vehicle class with displayInfo() method overridden in Car subclass to provide info about car company , model , price, seating and petrol.

**CODE:**

class Vehicle {

    void displayinfo() {

        System.out.println("This is a vehicle");

    }

}

class Car extends Vehicle {

    String Name, Model, Capacity;

    int Price;

    boolean Petrol;

    Car(String Name, String Model, String Capacity, int Price, boolean Petrol) {

        this.Name = Name;

        this.Model = Model;

        this.Capacity = Capacity;

        this.Price = Price;

        this.Petrol = Petrol;

    }

    @Override

    public void displayinfo() {

        System.out.println("Car name is: " + Name);

        System.out.println("The model is: " + Model);

        System.out.println("The price of the car is: " + Price);

        System.out.println("The seating capacity of the car is: " + Capacity);

        System.out.println("Is it petrol? " + Petrol);

    }

}

 class Mainq3 {

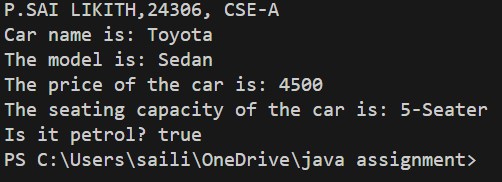
    public static void main(String[] args) {System.out.println("P.SAI LIKITH,24306, CSE-A");

        Car c1 = new Car("Toyota", "Sedan", "5-Seater", 4500, true);

        c1.displayinfo();

    }}

**OUTPUT:**

****

**ERRORS:-**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

CLASS DIAGRAM:-

|  |
| --- |
| Vehicle |
| +displayInfo(): void |

|  |
| --- |
| car |
| +displayInfo(): void |

2) A college is developing automated admission system that verifies students eligibility for UG and PG programs .Each program has different eligibility criteria based on the students percentage in their previous qualification.

* UG admission require minimum 60%
* PG admission require minimum 70%

CODE:

import java.util.Scanner;

class College {

    String name;

    int qualification;

    int percentage;

    // Constructor

    College(String name, int qualification, int percentage) {

        this.name = name;

        this.qualification = qualification;

        this.percentage = percentage;

    }

    // Default Eligibility method

    public void Eligibility() {

        System.out.println("Name: " + name + ", Qualification: " + qualification + ", Percentage: " + percentage);

        System.out.println("The candidate is a fluke");

    }

}

class UG extends College {

    UG(String name, int qualification, int percentage) {

        super(name, qualification, percentage);

    }

    @Override

    public void Eligibility() {

        System.out.println("Name: " + name + ", Qualification: " + qualification + ", Percentage: " + percentage);

        System.out.println("The candidate is eligible for UG");

    }

}

class PG extends College {

    PG(String name, int qualification, int percentage) {

        super(name, qualification, percentage);

    }

    @Override

    public void Eligibility() {

System.out.println("Name: " + name + ", Qualification: " + qualification + ", Percentage: " + percentage);

        System.out.println("The candidate is eligible for PG");

    }

}

public class Mainq4 {

    public static void main(String[] args) {

        System.out.println("P.SAI LIKITH,24306, CSE-A");

        Scanner input = new Scanner(System.in);

        // Taking inputs

        System.out.println("Enter your name:");

        String name = input.nextLine();

        System.out.println("Enter your qualification (e.g., 12 for high school, 10 for 10th, etc.):");

        int qualification = input.nextInt();

        System.out.println("Enter your percentage:");

        int percentage = input.nextInt();

        // Close scanner

        input.close();

        // Logic to check eligibility

        College candidate;

        if (percentage >= 70) {

            candidate = new PG(name, qualification, percentage);

        } else if (percentage >= 60) {

            candidate = new UG(name, qualification, percentage);

        } else {

            candidate = new College(name, qualification, percentage);

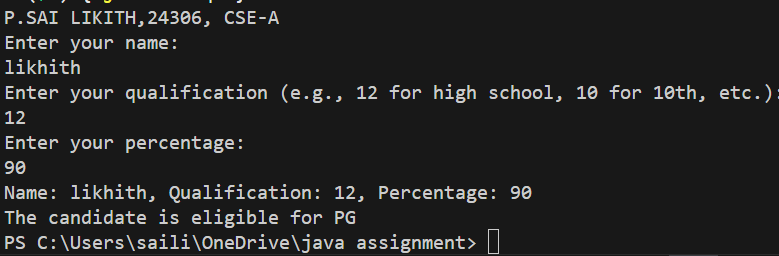
        }

        candidate.Eligibility();

    }

}

**OUTPUT:**



**Class Diagram:-**

|  |
| --- |
| **adm** |
| elg():void |

|  |  |
| --- | --- |
| ug | pg |
| +elg():void | +elg():void |

**ERROR:-**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

3) . Create a calculator class with overloaded methods to perform additions

1.add two integers

2.add two double values

3.add three integers

**CODE:**

class Calc{

    public int add(int a,int b){

        return a+b;

    }

    public double add(double a,double b){

        return a+b;

    }

    public int add(int a,int b,int c){

        return a+b+c;

    }

}

class Calc1{

    public static void main(String[] args){

        System.out.println("P.SAI LIKITH,24306, CSE-A");

        Calc C1=new Calc();

        System.out.println("Sum of 6 and 9 is: "+C1.add(6,9));

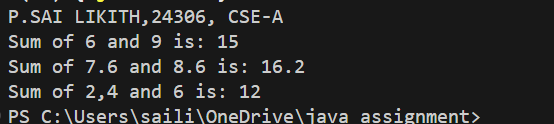
        System.out.println("Sum of 7.6 and 8.6 is: "+C1.add(7.6,8.6));

        System.out.println("Sum of 2,4 and 6 is: "+C1.add(2,4,6));

    }

}

**OUTPUT:**

****

CLASS DIAGRAM:-

|  |
| --- |
| cal |
| +add(int a,int b):int  +add(double a,double b):double  +add(int a,int b,int c):int |

ERRORS:-

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

4) . Create a shape class with method calculateArea() that is overloaded for different shapes (eg: square, rectangle).Then create a subclass Circle that overrides calculateArea() method for Circle.

**CODE:**

public class Shape {

    public double areaOfSquare(double side) {

        return side \* side;

    }

    public double areaOfRectangle(double length, double width) {

        return length \* width;

    }

    public double areaOfCircle(double radius) {

        return 3.14 \* radius \* radius;

    }

    public static void main(String[] args) {

        System.out.println("P.SAI LIKITH,24306, CSE-A");

        Shape shape = new Shape();

        double square = shape.areaOfSquare(5);

        System.out.println("Area of a square: " + square);

        double rectangle = shape.areaOfRectangle(10, 20);

        System.out.println("Area of a rectangle: " + rectangle);

        double circle = shape.areaOfCircle(12);

        System.out.println("Area of a circle: " + circle);

    }

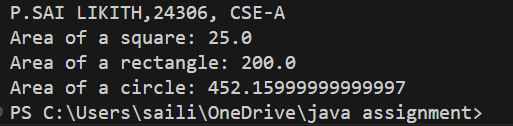
}

**OUTPUT:**

**CLASS DIAGRAM**:-

|  |
| --- |
| shape |
| +calarea(float side):float  +calarea(float l,float b):float  +calarea(float c):float |

|  |
| --- |
| Circle |
| +calarea(double r):double |

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**WEEK7**

**AIM:1**Write a java program to create an abstract class animal with an abstract method called sound create subclass lion and tiger that extend the animal class and implement the sound method to make a specific sound for each animal

**CODE:**

abstract  class Animal {

    abstract void Sound();

}

class lion extends Animal {

    public void Sound(){

        System.out.println("lRoar");

    }}

class Tiger extends Animal {

    public void Sound(){

        System.out.println("tRoar");

          }}

class week7q1 {

    public static void main(String[] args){

        System.out.println("NAME: P SAI LIKHITH ,ROLL NO :AV.SC.U4CSE24306,SEC:CSE-A");

lion l =new lion();

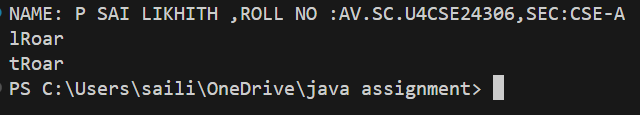
l.Sound();

Tiger t = new Tiger();

t.Sound();

    }}

**OUTPUT:**

****

**IMPORTANT POINTS:**

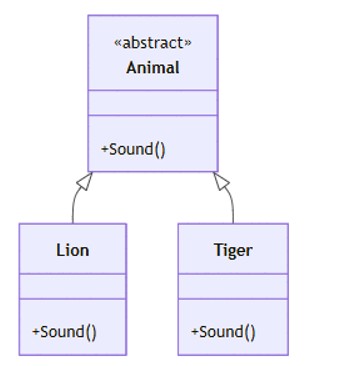
1. Abstract Class

* Animal is an abstract class.
* Abstract classes cannot be instantiated directly.
* It can have abstract methods (methods without a body) which must be overridden in child classes

2. Inheritance

* Lion and Tiger inherit from Animal using the extends keyword.
* They must override the Sound() method of the Animal class.

**CLASS DIAGRAM**



**AIM:2**: Write a java program to create an abstract class shape3D with abstract method calculatevolume () and calculatesurfacearea() create subclass sphere and cube that extends the shape3D class and implement the respective methods to calculate the volume and surface area of each shape.

**CODE:**

abstract class Shape3D {

    abstract double calculateVolume();

    abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

    private double radius;

    Sphere(double radius) {

        this.radius = radius;

    }

    @Override

    double calculateVolume() {

        return (4.0 / 3.0) \* Math.PI \* radius \* radius \* radius;

    }

    @Override

    double calculateSurfaceArea() {

        return 4 \* Math.PI \* radius \* radius;

    }

}

class  Cube extends Shape3D {

    private double side;

    Cube(double side) {

        this.side = side;

    }

    @Override

    double calculateVolume() {

        return side \* side \* side;

    }

    @Override

    double calculateSurfaceArea() {

        return 6 \* side \* side;

    }

}

class week7q2 {

    public static void main(String[] args) {

        System.out.println("NAME: P SAI LIKHITH,ROLL NO :AV.SC.U4CSE24306,SEC:CSE-A");

        Sphere s = new Sphere(2.3);

        System.out.println("Volume of Sphere: " + s.calculateVolume());

        System.out.println("Surface Area of Sphere: " + s.calculateSurfaceArea());

        Cube c = new Cube(2.3);

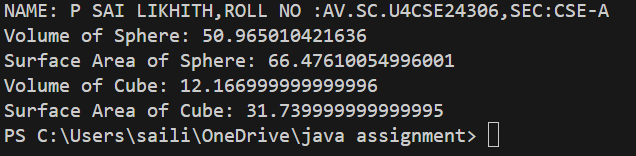
        System.out.println("Volume of Cube: " + c.calculateVolume());

        System.out.println("Surface Area of Cube: " + c.calculateSurfaceArea());

    }

}

**OUTPUT:**

**  
IMPORTANT POINTS:**

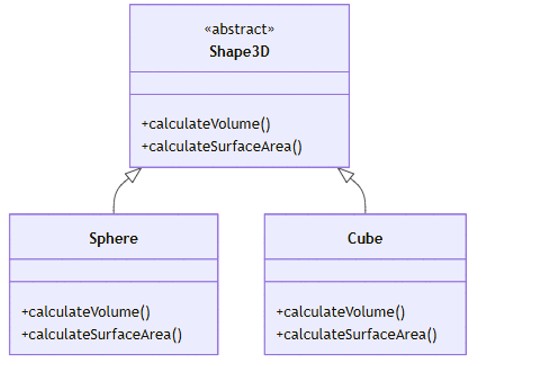
1. Abstract Class

* Shape3D is an abstract class.
* It defines two abstract methods: calculateVolume() and calculateSurfaceArea().
* Child classes must implement these methods.

2. Inheritance

* Sphere and Cube classes inherit (extends) from the Shape3D abstract class.
* They inherit the structure but provide their own specific calculations.

**CLASS DIAGRAM**



**AIM:3** write a java program using an abstract class to define a method for pattern printing.

Create an abstract class named pattern printer with an abstract method print pattern (int n) and a concrete method to display the pattern title

Implement two subclass:

* + - * 1. Star pattern: Print a right-angle triangle of stars(\*) .
        2. Number pattern : Print a right-angled triangle of increasing numbers.

In the main ()method, create objects of both subclass and print the pattern of given number of rows

**CODE:**

abstract class PatternPrinter {

    abstract void printPattern(int n);

    void displayTitle(String title) {

        System.out.println("" + title);

        System.out.println("");

    }

  }

  class StarPattern extends PatternPrinter {

    void printPattern(int n) {

        for (int i = 1; i <= n; i++) {

            for (int j = 1; j <= i; j++) {

                System.out.print("\* ");

            }

            System.out.println();

        }

    }

  }

  class NumberPattern extends PatternPrinter {

    void printPattern(int n) {

        int number = 1;

        for (int i = 1; i <= n; i++) {

            for (int j = 1; j <= i; j++) {

                System.out.print(number + " ");

                number++;

            }

            System.out.println();

        }

    }

  }

  class week7q3 {

    public static void main(String[] args) {

      System.out.println("NAME:P SAI LIKHITH , ROLL NO:24306,SEC:CSE-A ");

        int rows = 5;

        StarPattern star = new StarPattern();

        star.displayTitle("Star Pattern");

        star.printPattern(rows);

        NumberPattern number = new NumberPattern();

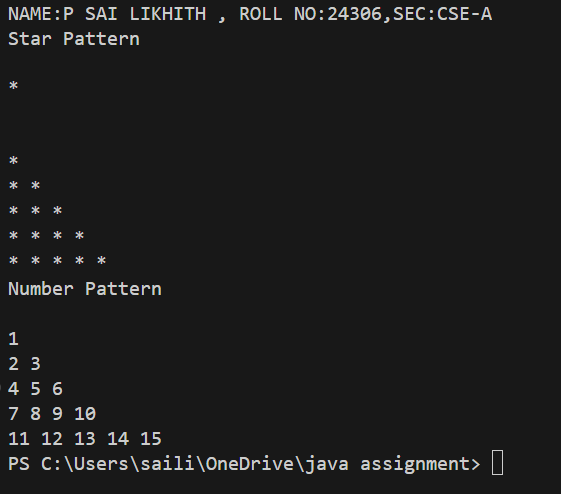
        number.displayTitle("Number Pattern");

        number.printPattern(rows);

    }

  }

**OUTPUT:**

****

**IMPORTANT POINTS:**

1. Abstract Class

* PatternPrinter is an abstract class.
* It has one abstract method printPattern(int n) — must be overridden in child classes.
* It also has a normal method displayTitle(String title) which prints the title before the pattern.

2. Inheritance

* StarPattern and NumberPattern inherit from PatternPrinter.
* They override printPattern() with their own way of printing the pattern.

**WEEK8**

**AIM1:** Write a java program to create an interface shape with the getperimeter() method.Create 3classes rectangle,circle and triangle that implements shape interface.Implement getperimeter() method for each of 3classes.

**CODE:**

interface Main1{

double getPerimeter( );

}

class Rectangle implements Main1 {

private double length;

private double width;

Rectangle(double length, double width){

this.length = length;

this.width = width;

}

public double getPerimeter() {

return 2 \* (length + width);

}

}

class Circle implements Main1 {

private double radius;

Circle(double radius) {

this.radius = radius;

}

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

}

class Triangle implements Main1 {

private double side1, side2, side3;

Triangle(double side1, double side2, double side3){

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

public double getPerimeter(){

return side1 + side2 + side3;

}

}public class main22{

public static void main(String[] args) {

System.out.println("NAME: P SAI LIKHITH ROLL NO: 24306 SEC: CSE-A");

Main1 rectangle = new Rectangle(3,4);

Main1 circle = new Circle(5);

Main1 triangle = new Triangle(2,5,6);

System.out.println("Rectangle perimeter: " + rectangle.getPerimeter());

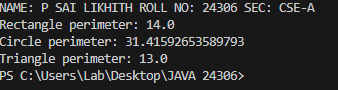
System.out.println("Circle perimeter: " + circle.getPerimeter());

System.out.println("Triangle perimeter: " + triangle.getPerimeter());

}

}

**OUTPUT**

****

**ERRORS**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Error** | **Rectification** |
| **1** | **Compilation error** | **Add correct return type** |
| **2** | **Access modifier issues(OOP error)** | **Change the modifier or add getter/setter** |

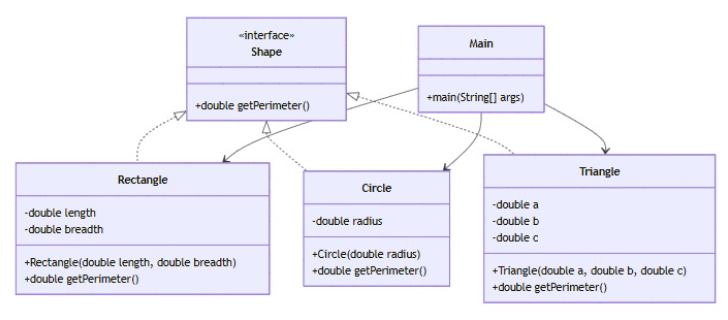
**IMPORTANT POINTS**

Getter and setter :

A getter method is used to retrieve the value of a private variable

A setter method is used to set or update the value of a private variable**.**

**CLASS DIAGRAM**



**AIM2:** Write a java program to create an interface label with method play that takes no arguments and return void create 3 classes football,volley ball and basketball that implements interface and override play() method to play respective sports.

**CODE**

interface Playable {

void play();

}

class Football implements Playable {

public void play() {

System.out.println("Playing football");

}

}

class Volleyball implements Playable {

public void play() {

System.out.println("Playing volleyball");

}

}

class Basketball implements Playable {

public void play() {

System.out.println("Playing basketball");

}

}

public class week8q2 {

public static void main(String[] args) {

System.out.println("NAME: P SAI LIKHITH ROLL NO: 24306 SEC: CSE-A" );

Playable football = new Football();

Playable volleyball = new Volleyball();

Playable basketball = new Basketball();

football.play();

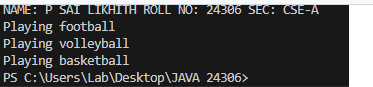
volleyball.play();

basketball.play();

}

}

**OUTPUT**



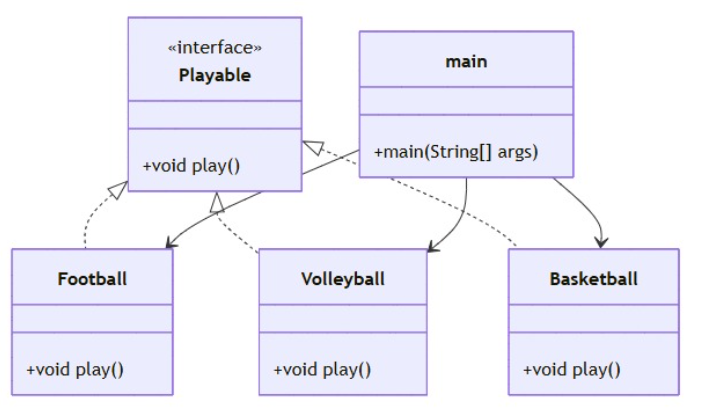
**ERRORS**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Error** | **Rectification** |
| **1.** | **Football is not abstract and does not override abstract method play()** | **Implement the play method from interface** |
| **2.** | **Playables.java but public class play test** | **Class play test is public,should be declared in a file named playtest.java** |

**IMPORTANT POINTS**

* The Playable interface defines the contract (play() method).
* Each class (Football, Volleyball, Basketball) implements the interface and provides a specific behavior for play().
* This demonstrates polymorphism using interfaces in Java**.**

**CLASS DIAGRAM**

**AIM3**: Write a java program to implement a login system using interfaces

**CODE**

import java.util.Scanner;

interface Login {

boolean authenticate(String username, String password);

}

class UserLogin implements Login {

private final String validUsername = "admin";

private final String validPassword = "password123";

@Override

public boolean authenticate(String username, String password) {

return validUsername.equals(username) && validPassword.equals(password);

}

}

public class LoginSystem {

public static void main(String[] args) {

System.out.println("NAME: P SAI LIKHITH ROLL NO: 24306 SEC: CSE-A");

Scanner scanner = new Scanner(System.in);

UserLogin userLogin = new UserLogin();

System.out.print("Enter username: ");

String username = scanner.nextLine();

System.out.print("Enter password: ");

String password = scanner.nextLine();

if (userLogin.authenticate(username, password)) {

System.out.println("Login successful!");

} else {

System.out.println("Login failed! Invalid username or password.");

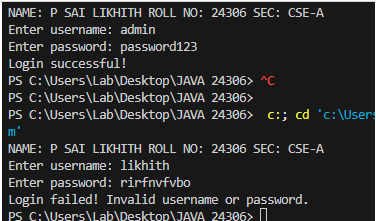
}

scanner.close();

}

}

**OUTPUT**

**ERRORS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error** | **Error rectification** |
| **1** | **Login\_System() not**  **overridden in University classes.** | **Implement Login\_System() in**  **university class using @override annotation** |
| **2** | **Error: ”;” expected** | **Giving the ‘;’ in wanted places** |

**IMPORTANT POINTS**

Boolean type is used in many applications when we need to check the input is true or false**.**

**CLASS DIAGRAM**

